Amendments to the Claims

Please amend claims 1, 3, 5, 12, 14, 16, 23, 25, 27 & 29 as set forth below, and cancel claims 10, 11, 21, 22, 34 & 35 without prejudice. In accordance with current amendment practice, all pending claims are reproduced below. The changes in the amended claims are shown by underlining (for added matter) and strikethrough/double brackets (for deleted matter).

1. (Currently Amended) A method of evaluating a software component, comprising:

deriving a conceptual layering scheme of the software component, the conceptual layering scheme dividing the software component into multiple layers, the multiple layers taking into account relationships that exist between layers;

providing an abstraction matrix that describes the multiple layers of the software component and accounts for the relationships that exist between the layers, the abstraction matrix further comprising at least one test case scenario for each layer and mapped expected results ~~therefore~~ therefor;

testing the multiple layers of the software component using the test ~~cases~~ case scenarios to generate test results; ~~and~~

evaluating the test results using the abstraction matrix, the evaluating including for each layer comparing a test case employed in the testing of the layer to the at least one test case scenario of the abstraction matrix therefor, and if a match is found, comparing the test result for that test case with the mapped expected result ~~therefore~~ for that test case scenario in the abstraction matrix; and

wherein evaluation of the software component is accomplished upon completion of evaluation of the test results.

2. (Original) The method of claim 1, wherein the evaluating comprises automatically evaluating the test results using the abstraction matrix.

3. (Currently Amended) The method of claim 1, further comprising, prior to said testing, deriving at least some test cases for each layer from the at least one test case scenario of the abstraction matrix.

4. (Original) The method of claim 1, wherein the software component comprises multiple states, and wherein the abstraction matrix comprises multiple test case scenarios, each test case scenario being associated with a different state of the software component.

5. (Currently Amended) The method of claim 1, further comprising generating an error log, said error log containing a list of test cases each having a test result which failed to match the mapped expected result ~~therefore~~ therefor in a matching test case scenario of the abstraction matrix.

6. (Original) The method of claim 5, wherein said error log further includes the test results of the failing test cases, as well as the mapped expected results from the abstraction matrix for the failing test cases.

7. (Original) The method of claim 1, further comprising creating at least one test results file and at least one abstraction matrix file, and wherein said evaluating comprises automatically reviewing each test result in the at least one test results file by comparing its test case to the at least one test case scenario of the abstraction matrix contained in the at least one abstraction matrix file.

8. (Original) The method of claim 1, wherein the providing comprises creating the abstraction matrix from a functional specification of the software component.

9. (Original) The method of claim 1, wherein the evaluating comprises automatically extracting test statistics, said test statistics including a total number of test cases executed, and at least one of a total number of test cases successfully executed or a total number of test cases unsuccessfully executed.

10. Canceled.

11. Canceled.

12. (Currently Amended) A system for evaluating a software component comprising:

means for deriving a conceptual layering scheme of the software component, the conceptual layering scheme dividing the software component into multiple layers, the multiple layers taking into account relationships that exist between layers;

an abstraction matrix that describes the multiple layers of the software component and accounts for the relationships that exist between layers, the abstraction matrix further comprising at least one test case scenario for each layer and mapped expected results ~~therefore~~ therefor;

means for testing the multiple layers of the software component using the test ~~cases~~ case scenarios to generate test results; ~~and~~

means for evaluating the test results using the abstraction matrix, the means for evaluating including for each layer means for comparing a test case employed in the testing of the layer to the at least one test case scenario of the abstraction matrix therefor, and if a match is found, for comparing the test result for that test case with the mapped expected result ~~therefore~~ for that test case scenario in the abstraction matrix; and

wherein evaluation of the software component is accomplished upon completion of the evaluation of the test results.

13. (Original) The system of claim 12, where the means for evaluating comprises means for automatically evaluating the test results using the abstraction matrix.

14. (Currently Amended) The system of claim 12, further comprising means for deriving at least some test cases for each layer from the at least one test case scenario of the abstraction matrix for use by the means for testing.

15. (Original) The system of claim 12, wherein the software component comprises multiple states, and wherein the abstraction matrix comprises multiple test case scenarios, each test case scenario being associated with a different state of the software component.

16. (Currently Amended) The system of claim 12, further comprising means for generating an error log, said error log containing a list of test cases each having a test result which failed to match the mapped expected result ~~therefore~~ therefor in a matching test case scenario of the abstraction matrix.

17. (Original) The system of claim 16, wherein said error log further includes test results of the failing test cases, as well as mapped expected results from the abstraction matrix for the failing test cases.

18. (Original) The system of claim 12, further comprising means for creating at least one test results file and at least one abstraction matrix file, and wherein said means for evaluating comprises means for automatically reviewing each test result in the at least one test results file by comparing its test case to the at least one test case scenario of the abstraction matrix contained in the at least one abstraction matrix file.

19. (Original) The system of claim 12, wherein the means for providing comprises means for creating the abstraction matrix from a functional specification of the software component.

20. (Original) The system of claim 12, wherein the means for evaluating comprises means for automatically extracting test statistics, said test statistics including a total number of test cases executed, and at least one of a total number of test cases successfully executed or a total number of test cases unsuccessfully executed.

21. Canceled.

22. Canceled.

23. (Currently Amended) A system for evaluating a software component comprising:

means for deriving a conceptual layering scheme of the software component, the conceptual layering scheme dividing the software component into multiple layers, the multiple layers taking into account relationships that exist between layers;

an abstraction matrix that describes the multiple layers of the software component and accounts for the relationships that exist between the layers, the abstraction matrix further comprising at least one test case scenario for each layer and mapped expected results ~~therefore~~ therefor;

a first computing unit adapted to test the multiple layers of the software component using the test ~~cases~~ case scenarios to generate test results; ~~and~~

a second computing unit adapted to evaluate the test results using the abstraction matrix, the evaluating including for each layer comparing a test case employed in the testing of the layer to the at least one test case scenario of the abstraction matrix therefor, and if a match is found, comparing the test result for that test case with the mapped expected result ~~therefore~~ for that test case scenario in the abstraction matrix; and

wherein evaluation of the software component is accomplished upon completion of the evaluation of the test results.

24. (Original) The system of claim 23, wherein the first computing unit and the second computing unit comprise a single computing unit.

25. (Currently Amended) At least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform a method of evaluating a software component, comprising:

deriving a conceptual layering scheme of the software component, the conceptual layering scheme dividing the software component into multiple layers, the multiple layers taking into account relationships that exist between layers;

providing an abstraction matrix that describes the multiple layers of the software component and accounts for the relationships that exist between the layers, the abstraction matrix further comprising at least one test case scenario for each layer and mapped expected results ~~therefore~~ therefor;

testing the multiple layers of the software component using the test ~~cases~~ case scenarios to generate test results; ~~and~~

evaluating the test results using the abstraction matrix, the evaluating including for each layer comparing a test case employed in the testing of the layer to the at least one test case scenario of the abstraction matrix therefor, and if a match is found, comparing the test result for that test case with the mapped expected result ~~therefore~~ for that test case scenario in the abstraction matrix; and

wherein evaluation of the software component is accomplished upon completion of the evaluation of the test results.

26. (Original) The at least one program storage device of claim 25, where the evaluating comprises automatically evaluating the test results using the abstraction matrix.

27. (Currently Amended) The at least one program storage device of claim 25, further comprising, prior to said testing, deriving at least some test cases for each layer from the at least one test case scenario of the abstraction matrix.

28. (Original) The at least one program storage device of claim 25, wherein the software component comprises multiple states, and wherein the abstraction matrix comprises multiple test case scenarios, each test case scenario being associated with a different state of the software component.

29. (Currently Amended) The at least one program storage device of claim 25, further comprising generating an error log, said error log containing a list of test cases each having a test result which failed to match the mapped expected result ~~therefore~~ therefor in a matching test case scenario of the abstraction matrix.

30. (Original) The at least one program storage device of claim 29, wherein said error log further includes test results of the failing test cases, as well as mapped expected results from the abstraction matrix for the failing test cases.

31. (Original) The at least one program storage device of claim 25, further comprising creating at least one test results file and at least one abstraction matrix file, and wherein said evaluating comprises automatically reviewing each test result in the at least one test results file by comparing its test case to the at least one test case scenario of the abstraction matrix contained in the at least one abstraction matrix file.

32. (Original) The at least one program storage device of claim 25, wherein the providing comprises creating the abstraction matrix from a functional specification of the software component.

33. (Original) The at least one program storage device of claim 25, wherein the evaluating comprises automatically extracting test statistics, said test statistics including a total number of test cases executed, and at least one of a total number of test cases successfully executed or a total number of test cases unsuccessfully executed.

34. Canceled.

35. Canceled.

* * * * *